**ЛАБОРАТОРНАЯ РОБОТА №1**

**ОЗНАКОМЛЕНИЕ С ОСНОВНИМИ ОСОБЕННОСТЯМИ**

**ЯЗЫКА С# .NET**

**Цель:** Ознакомление с основными особенностями языка С# .NET.

**Вариант 5. Счет мобильного оператора**

В проекте предусмотреть разделение интерфейса и логики типов, которые создавались во время выполнения задания. Предусмотреть создание исключений во время введения некорректных значений.

**Данные типа:**

* название тарифа: определяется перечислением из 3-4 состояний;
* стоимость минуты разговора;
* текущая сума на счету;
* номер телефона: содержит 10 цифр;
* массив подключения услуги: определяется перечислением.

**Методы типа:**

* конструктор по умолчанию;
* инициализирующий конструктор;
* переход на новый тарифный план: тариф является аргументом;
* звонок: входные аргументы – количество минут и второй номер, выходное значение – сумма, списанная со счета за проведенный разговор;
* подключение/отключение услуги: услуга имеет название и цену;
* пополнение счета;
* получение текущего состояния счета/истории звонков.

**Код:**

Service.cs

namespace Lab1.Enums

{

enum Service

{

FirstService,

SecondService,

ThirdService,

FourthService

}

}

TariffName.cs

namespace Lab1.Enums

{

enum TariffName

{

FirstTariff,

SecondTariff,

ThirdTariff,

FourthTariff

}

}

MobileAccount.cs

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

using Lab1.Enums;

namespace Lab1

{

class MobileAccount

{

private TariffName \_tariffName;

private double \_pricePerMin;

private double \_balance;

private string \_phoneNumber;

private readonly string phoneNumberRegex = @"^\d{10}$";

private List<Service> \_services;

private List<string> \_callHistory;

//Конструкор по умолчанию

public MobileAccount()

{

\_services = new List<Service>();

\_callHistory = new List<string>();

}

//Инициализирующий конструктор

public MobileAccount(TariffName tariffName, double pricePerMin, int balance, string phoneNumber) : this()

{

\_tariffName = tariffName;

\_pricePerMin = pricePerMin;

\_balance = balance;

if (Regex.IsMatch(phoneNumber, phoneNumberRegex))

\_phoneNumber = phoneNumber;

//Генерация исключения в случае невалидных данных

else throw new Exception("Invalid phone number");

}

//Метод для установки нового тарифа

public void SetNewTariff(TariffName tariff)

{

\_tariffName = tariff;

Console.WriteLine($"Для абонента {this.\_phoneNumber} установлен тариф: {this.\_tariffName}");

}

//Метод для совершения звонка

public double Call(int minuteCount, string phoneNumber)

{

double callPrice = 0;

//Валидация введенного номера

if (\_phoneNumber != phoneNumber && Regex.IsMatch(phoneNumber, phoneNumberRegex))

{

//Расчет цены звонка

callPrice = \_pricePerMin \* minuteCount;

}

//Генерация исключения в случае невалидных данных

else throw new Exception("Call error");

\_balance -= callPrice;

\_callHistory.Add(phoneNumber);

Console.WriteLine($"Абонент {this.\_phoneNumber} совершил звонок на номер {phoneNumber}, на сумму {callPrice}");

return callPrice;

}

//Подключение нового сервиса для абонента

public void AddService(Service service, double servicePrice)

{

\_services.Add(service);

\_balance -= servicePrice;

Console.WriteLine($"Для абонента {this.\_phoneNumber} добавлен сервис {service} за {servicePrice}");

}

//Удаление сервиса для абонента

public void RemoveService(Service service)

{

\_services.Remove(service);

Console.WriteLine($"Для абонента {this.\_phoneNumber} удален сервис {service}");

}

//Пополнение счета

public void AddMoney(double addedSum)

{

if (addedSum > 0)

\_balance += addedSum;

Console.WriteLine($"На счет абонента {this.\_phoneNumber} начислено {addedSum}");

}

//Получение текущей суммы на счету

public double GetBalance()

{

return \_balance;

}

//Получение истории звонков

public List<string> GetCallHistory()

{

return \_callHistory;

}

//Вывод информации об абоненте

public void ShowAccountInfo()

{

Console.WriteLine(

"\nНазвание Тарифа: {0}\nСтоимость минуты: {1}\nТекущая сумма на счету: {2}\nНомер телефона: {3}\n",

\_tariffName, \_pricePerMin, \_balance, \_phoneNumber);

Console.WriteLine("Подключенные сервисы:");

\_services.ForEach(service => Console.WriteLine(service));

Console.WriteLine("История звонков:");

\_callHistory.ForEach(call => Console.WriteLine(call));

Console.WriteLine();

}

}

}

Program.cs

using System;

using Lab1.Enums;

namespace Lab1

{

class Program

{

static void Main(string[] args)

{

//Создание обьекта аккаунта

var mobileAccount = new MobileAccount(TariffName.FirstTariff, 2.55, 1000, "1234567890");

mobileAccount.Call(50, "3548965441");

mobileAccount.Call(25, "5548765441");

mobileAccount.AddService(Service.FirstService, 100);

mobileAccount.AddService(Service.FourthService, 400);

mobileAccount.ShowAccountInfo();

Console.Read();

}

}

}

**Результат:**

